多安定問題として見たディジタルネットワークの同期問題 (多値論理およびその応用)

AUTHOR(S):

阿江, 忠; 相原, 玲二; 天満, 尚二; 戸井, 哲也

数理解析研究所講究録
455巻 1982年 245-259

多安定問題として見たディジタル ネットワークの同期問題

広島大 工　　前江 忠　相原玲二
天満尚二　戸井哲也

1. まえがき

演算の基礎は論理演算により表されるが，演算の〝流れ〟を結合していく単位は値を保持するレジスタである。ハードウェア上で同期をとるシステムではシステムが1つのクロックで動作するから，〝値の転送〟は故障のない限り確実に行われるものとして扱われる。一方，ハードウェア上では同期をとる機構のないシステム（以下，非同期システムと呼ぶ）では〝値の転送〟が確実に行われるような保証をつける必要がある。VLSIをはじめとする複雑化したシステム（分散システム）は非同期システムとして扱われることが多い。

非同期システムの問題は古くから多くの研究がなされているが，筆者らは，この問題を

logical な考察
semantic な考察

に分類する。 logical な考察とは古くは Unger[1] や Muller のものがあり，最近では syntactic な議論が該当しよう。確かに最近の議論には数学的な進展を感じるが[2]，logical な考察では syntactic な議論が中心であるため，"値の転送の確実さ" という semantic なものの表現は難しい。一方，semantic な考察はネットワークや分散システムでのコンピュータ通信が盛んになってきて本格化してきたもので，今後ますます盛んになるものと思われる。通信については Dijkstra 流のもの[3][4] があるが，これは logical な議論に入る。むしろ，semantic な議論には OS (operating system) におけるプロセス同期問題が大いに参考になる。(むろん，ほとんどは単一プロセッサのものであるので，そのまま議論が適用できるという意味ではない。) このような分散システムにおける通信の問題を semantic に議論してきた1人として Lamport[5] があげられよう。一方，これら semantic な議論は，semantic であるがゆえに，論理的には複雑であるのが欠点である。[6] このような実情をふまえて，規模は小から大までの非同期分散システムを "ディジタルネットワーク" と称することにして

その同期問題について少し述べたいと思う。多安定問題はその背景にあるという意味で"多安定問題として見た"というただし書きをつけたが，それが成功したという意味ではなく，本稿では問題を提起するというぐらいの気持である。

2. ディジタルネットワークモデル

システムが正常に動作しない一つにデッドロックがあるが，point to point communication をベースとするネットワークでのデッドロックは，一般によく論じられる資源共有の際のデッドロックより一段低いレベルの"ハンドシェイク失敗"という類のものもある。このあたりを明確にするためディジタルネットワークにおけるデッドロック全般を整理すると次のような分類ができる。

ディジタルネットワークにおけるデッドロック問題

(i) 論理レベル

(ii) 通信レベル

(iii) 資源レベル

(i) → (iii) の順にレベルは高くなる。もっとも低い (i) の論理レベルは非同期順序回路の設計の際に生じるもので，論理設計の誤りがその因となる。このレベルの議論は2値論理がふつう用いられるが，3値論理や Fuzzy Logic も導入

されている。逆に一番高い(iii)の資源レベルは通常ネットワークのデッドロック問題としてよく議論されているが[7][8]，semanticな色彩が強くなる。中間の(ii)の通信レベルには，パケット交換の高級なものをとりあげると複雑な問題になるが[9][10]，point to point communicationそのものは比較的問題の少ないレベルである。というのは，まったくsyntacticに解釈してしまうと(i)の論理レベルと同じものとなる。しかし，ここではこのレベルをsemantic に解釈して議論する。そのために，次のようなノードが3状態をもつネットワークモデルを導入する。

ディジタルネットワークをノードとラインによるグラフにより表現するとき，ノードの動作として

0状態 (process状態) passive
1状態 (drive状態) } active
2状態 (driven状態) }

の3状態があるものとする。0のprocess状態とはノードが処理状態にあるときをいう。また，idle状態もこれに含まれるとする。この状態では，他のノードとの関連がないという意味でpassiveである。これに対し，1と2の状態は他のノードと関連して動作するという意味でactiveな状態である。1は他のノードへ割込みをかけるという意味

で drive状態と呼び，逆に，2は他のノードから割込みをかけられるという意味で driven状態という。

なお，drive / driven の方向性の概念と"値の転送"が1方向 / 2方向ということとは直接関係がなく，driveの方向が1方向の流れであっても，値の転送は2方向になりうることに注意されたい。

この3状態モデルのノードは，隣のノードから次のように駆動される。

```
if   state = 0   and
        neighbourhood state = 1
then    state := 2
```

また駆動解除は

```
if   state = 2   and
        neighbourhood state ≠ 1
then    state := 0
```

driveを無制限に許すとデッドロックの危険性がある。コンピュータシステムのレベルでは動作が停止しないで続く（live）か停止する（dead）かが議論されたが，VLSIシステムのように規則性の高いシステムでは，もっと強い条件での動作，つまり，スムーズ（smooth）な動作を問題

にあることが多い。（とくにVLSI計算ではsystolic computationと呼ばれている。）

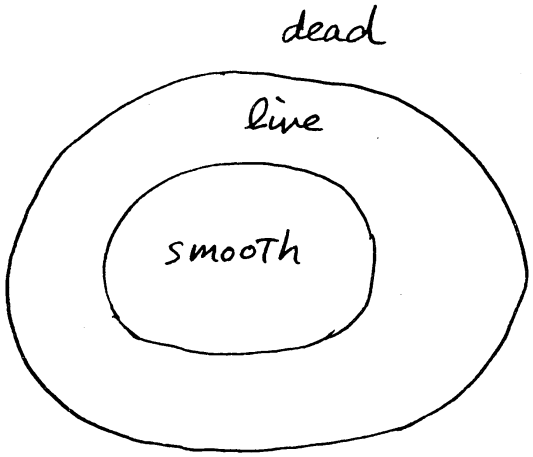


図1. 動作の分類

図1のように動作を分類したとき，deadは

デッドロック（dead lock）

餓死（starvation）

に分けられる。3状態ノード間の通信をsemanticにながあるとき，デッドロックの回避は容易にできる。

注．前述の"駆動"および"駆動解除"の方式を用いるとき，若干のタイミング差でデッドロックは回避できる。もっともデッドロック回避の方策を考える部分は3状態以外にあると想定している。これはETHERNETなどCSMA/CD方式の衝突回避の方策と同じようなものである。

以下では，分類としては"餓死"に入る deadの対策を考える。前述のパケット交換における dead lock はその一つである[9][10]。ここでは，交換方式まで立入らずに，"ハンドシェイク失敗"というレベルの議論の拡大を論じる。

3. 通信の確立が容易なネットワークの形状について

もっとも通信速度の速いネットワーク形状としてブロードキャストネットワーク[11]が知られているが，その一例は図2のようなスターポリゴンである。ただし，これまでの議論では通信の確実さは不問としてきている。"値の転送"をニ

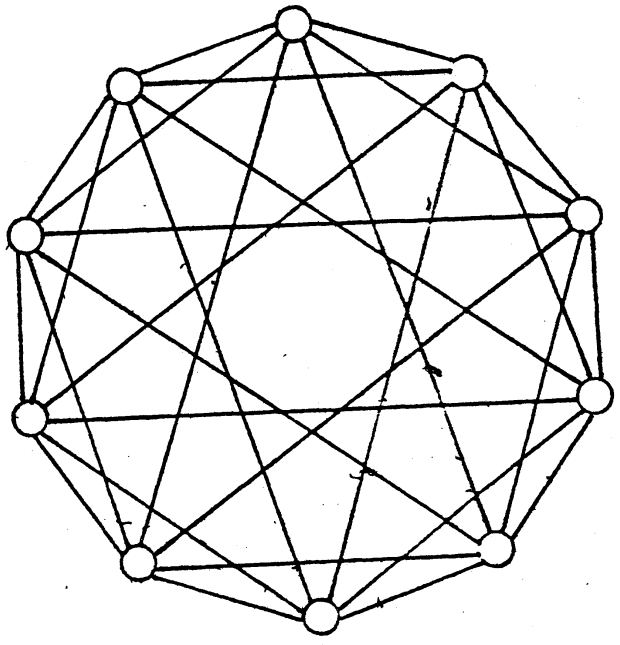


図2. スターポリゴンの例

こでは"通信"と呼ぶことにして，通信の確かさを保証する手段を次のように分類する。

(1) 既時確認

図3のように1状態（drive状態）にあるノードが2状態（driven状態）にあるノードへメッセージを送り（send），その答を受とる（receive）ことを通信のたびに行なう方法である。この方法はノード対ごとにsendとreceiveを行なうために時間はかかるが，確実な方法であり，コンピュータネットワークではふつうの手法である。

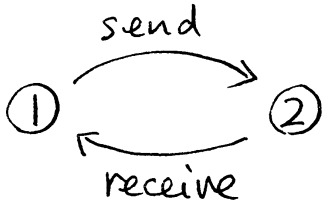


図3. 既時確認

(2) 一括確認

既時確認の欠点は小さなモジュールからなるネットワークでは手間がかかる点である。その改良は逐一receiveをとるのをやめ，経過ノードを経てきてから一括して確認する方法である。

試みに任意の1つのノードから，すべてのノードと同期のために通信するには，ブロードキャストネットワーク上で図4のようなブロードキャストフローをつくればよい。

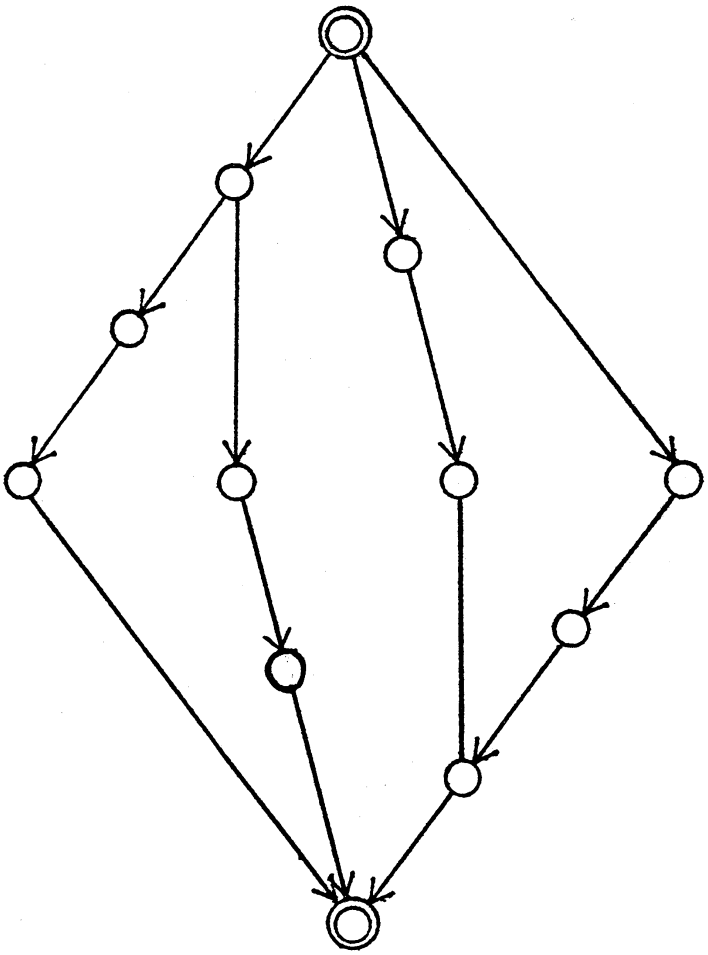
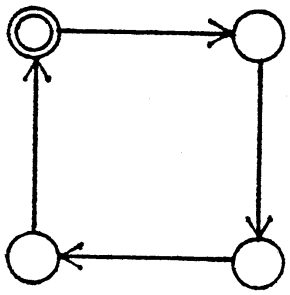


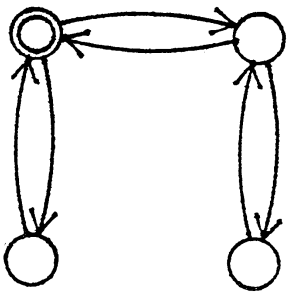
図4. ブロードキャストフローの例

その他のフローの例については文献(12)を参照されたい。一般には，既時確認と一括確認の混在がありえよう。

ノード数4の場合の既時確認と一括確認を図5に示す。



(a) 一括確認



(b) 既時確認

図5. 通信確認の方法

次に，通信の確立がノード対であるラインの（一時的なものを含む）不良のために上手くいかないときを考えると，予備の迂回路（detour）があると工合いがよい。

長さ2のdetourを shortest detour（sdと略す）と呼び，互いに素なsdがi個あるときisdと略す。また，グラフの連結度（connectivity）がkのときkCと略す。isd, kCグラフのうちi≦kのものを対象にk=3までのうち極小なグラフを2，3求めてみよう。

以下，ノード数をn，ライン数をeと記す。

(1) 1sd, 2Cグラフ

図6に構成法を示す。　$e = n + \lceil \frac{n}{2} \rceil$

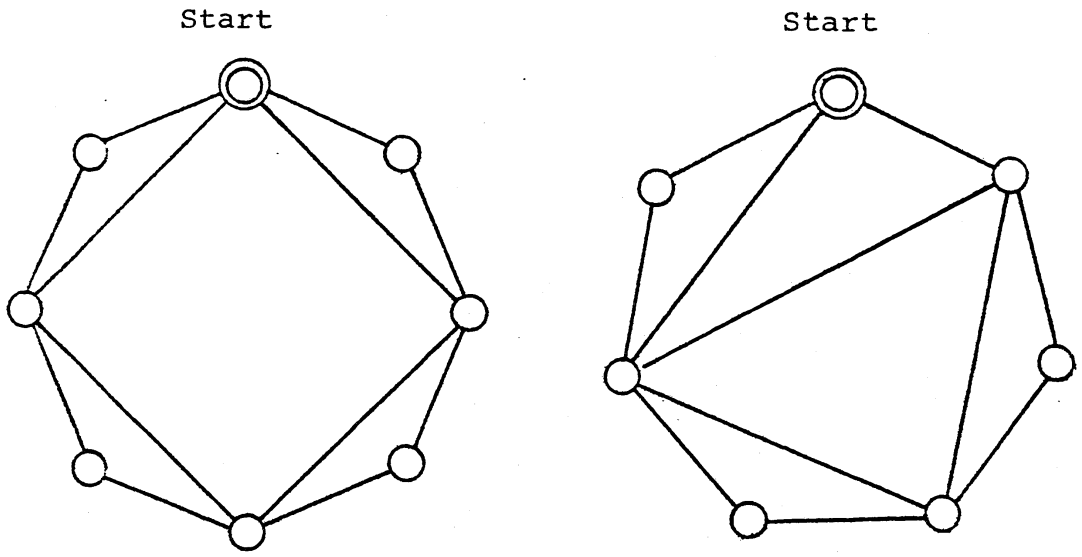


図6. 1sd, 2C グラフ

(2) 1sd, 3Cグラフ

図7に示すようなホイールが該当する。　$e = 2(n-1)$

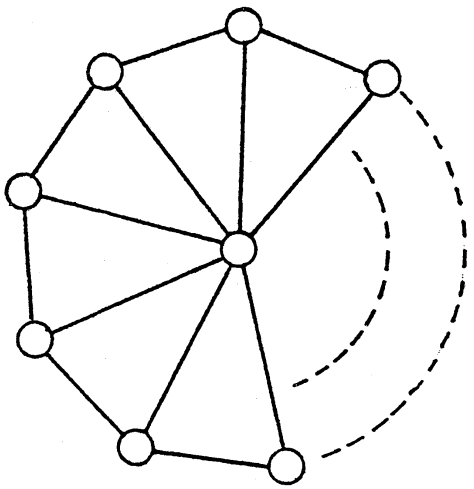


図7. 1sd, 3C グラフ

(3) 2sd, 3C グラフ

図8のような極大平面グラフが該当する。 $e = 3(n-2)$

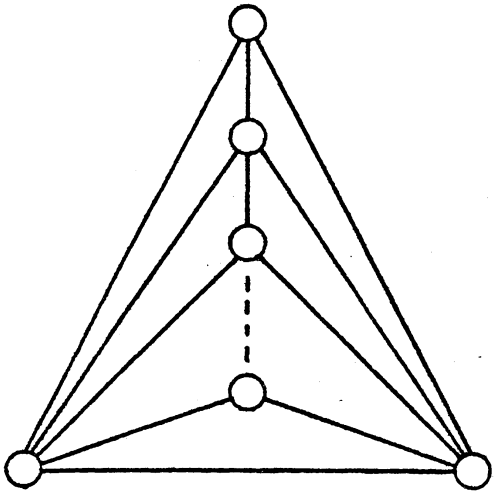


図8. 2sd, 3C グラフ

(4) 1sd, 2C 有向グラフ

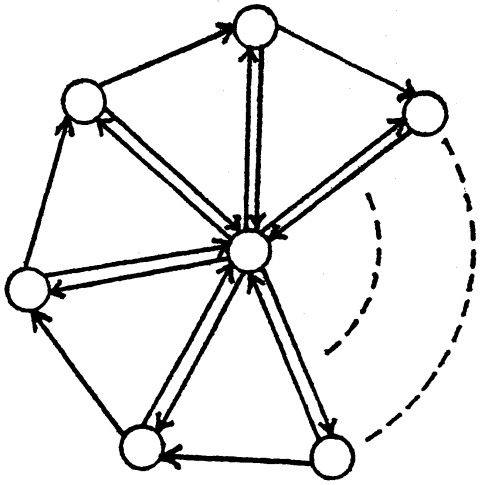


図9. 擬ホイール

図9のような擬ホイール（pseudo wheel）[13]が該当する。　有向ラインを1とすれば，$e = 3n - 3$ であるが，無向ラインに換算すれば，$e = (3n - 3)/2$ となる。このようなネットワーク結合の実例はすでに報告している[14]。

以上，う廻路をもつネットワークを具体例を示しつつ述べたが，一般的性質はまだよくわかっていない。

4．あとがき

ディジタルネットワークの同期問題について，若干の話題を提供した。最近の事情はマイクロプロセッサなどの急速な進歩により，事例の方が先行しているような感じが強い。まったく焦点の定まらない議論を述べたが，多少なりとも参考になれば幸いである。

文献

(1) S. H. Unger : "Asynchronous Sequential Switching Circuits", John Wiley and Sons. (1969)

(2) E. Arjomandi : "A difference in efficiency between synchronous and asynchronous systems" Proc. ACM STOC, pp. 128-132 (1981)

(3) E. W. Dijkstra : "Self-stabilizing Systems in spite of distributed control", CACM, 17, 11, pp. 643-644 (1974)

(4) C. Whitby-Strevens : "On the performance of Dijkstra's self-stabilising algorithms in spite of distributed control", Proc. 1st Intern. Conf. on Distributed Computing Systems, pp. 586-592 (1979)

(5). L. Lamport : "Time, clocks, and the ordering of events in a distributed system", CACM. 21, 7, pp. 558-565 (1978)

(6) J. Reif et al. : "Distributed algorithms for synchronizing interprocess communication within real time", Proc. ACM STOC, pp. 133-145 (1981)

(7) 杉山ほか "分散形システムにおけるデッドロックの検出と回復", 信学論(D), J63-D, 1, pp. 40-47 (1980)

(8) G. Ricart et al. : "An optimal algorithm for mutual exclusion in computer networks", CACM, 24, 1, pp. 9-17 (1981)

(9) S. Toueg et al. : "Deadlock-free packet switching networks", Proc. ACM STOC (1979)

(10) T. Kikuno et al. : "Local controllers for packet switching networks" Proc. IEEE FTCS, pp. 325-327 (1980)

(11) A. M. Farley : "Minimal broadcast network", Networks, 9, pp. 313-332 (1979)

(12) 阿江ほか："通信オーバヘッドを考慮した並列処理ネットワークにおけるフロー問題"，信学技報 AL81-31 (1981)

(13) 阿江ほか："ホイールネットワークの性質について"，信学技報 EC80-53 (1980)

(14) T. Ae et al.："A multiple microprocessor system with mutually diagnosing capability"，Proc. ICS (vol.I), pp. 375-388, Taipei (1980)